

```

; AltoIIMRT16K.mu
;
; last modified December 1, 1977 1:13 AM
;
; This is the part of the Memory Refresh Task which
; is specific to Alto IIs with Extended memory.
;
; Copyright Xerox Corporation 1979
$EngNumber      $30000;      ALTO II WITH EXTENDED MEMORY
;
; This version assumes MRTACT is cleared by BLOCK, not MAR< R37
; R37 [4-13] are the low bits of the TOD clock
; R37 [8-14] are the refresh address bits
; Each time MRT runs, four refresh addresses are generated, though
; R37 is incremented only once. Sprinkled throughout the execution
; of this code are the following operations having to do with refresh:
;
;   MAR< R37
;   R37< R37 +4          NOTE THAT R37 [14] DOES NOT CHANGE
;   MAR< R37 XOR 2       TOGGLES BIT 14
;   MAR< R37 XOR 200     TOGGLES BIT 8
;   MAR< R37 XOR 202     TOGGLES BITS 8 AND 14
;
MRT:   MAR< R37;          **FIRST REFRESH CYCLE**
      SINK< MOUSE, BUS;  MOUSE DATA IS AND'ED WITH 17B
MRTA:  L< T< -2, :TX0;   DISPATCH ON MOUSE CHANGE
TX0:   L< R37 AND NOT T, T< R37; INCREMENT CLOCK
      T< 3+T+1, SH=0;    IE. T< T +4. IS INTV TIMER ON?
      L< REFIIMSK AND T, :DOTIMER; [DOTIMER,NOTIMER] ZERO HIGH 4 BITS
NOTIMER: R37< L;        STORE UPDATED CLOCK
NOTIMERINT: T< 2;       NO STATE AT THIS POINT IN PUBLIC REGS
      MAR< R37 XOR T, T< R37; **SECOND REFRESH CYCLE**
      L< REFZERO AND T;    ONLY THE CLOCK BITS, PLEASE
      SH=0, TASK;         TEST FOR CLOCK OVERFLOW
      :NOCLK;             [NOCLK,CLOCK]
NOCLK: T< 200;
      MAR< R37 XOR T;      **THIRD REFRESH CYCLE**
      L< CURX, BLOCK;     CLEARS WAKEUP REQUEST FF
      T< 2 OR T, SH=0;    NEED TO CHECK CURSOR?
      MAR< R37 XOR T, :DOCUR; **FOURTH REFRESH CYCLE**
NOCUR: CURDATA< L, TASK;
MRTLST: CURDATA< L, :MRT;  END OF MAIN LOOP

DOTIMER: R37< L;        STORE UPDATED CLOCK
      MAR< EIALOC;       INTERVAL TIMER/EIA INTERFACE
      L< 2 AND T;
      SH=0, L< T< REFZERO.T; ***V3 CHANGE (USED TO BE BIAS)
      CURDATA< L, :SPCHK; CURDATA< CURRENT TIME WITHOUT CONTROL BITS

SPCHK: SINK< MD, BUS=0, TASK; CHECK FOR EIA LINE SPACING
SPIA:  :NOTIMERINT, CLOCKTEMP< L;

NOSPCHK: L< MD;         CHECK FOR TIME = NOW
      MAR< TRAPDISP-1;   CONTAINS TIME AT WHICH INTERRUPT SHOULD HAPPEN
      MTEMP< L;         IF INTERRUPT IS CAUSED,
      L< MD-T;         LINE STATE WILL BE STORED
      SH=0, TASK, L< MTEMP, :SPIA;

TIMERINT: MAR< ITQUAN;  STORE THE THING IN CLOCKTEMP AT ITQUAN
      L< CURDATA;
      R37< L;
      T< NWW;           AND CAUSE AN INTERRUPT ON THE CHANNELS
      MD< CLOCKTEMP;    SPECIFIED BY ITQUAN+1
      L< MD OR T, TASK;
      NWW< L, :NOTIMERINT;

;The rest of MRT, starting at the label CLOCK is unchanged

```